IN THE CLAIMS

Please enter the following amendments to claim 8, 16, 38 to overcome formality objections.

1. (Previously Amended) A computer-implemented method for a first component to invoke a second component asynchronously in an object-oriented computing environment, the computer-implemented method comprising:

receiving a request from a first component to invoke a second component;

maintaining the scope of the received request;

providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener for handling exceptions associated with the invocation of the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

2. (Original) The computer-implemented method of claim 1, wherein the request has a return type of void.

3. (Original) The computer-implemented method of claim 1, wherein the request is associated with no application specific exceptions.

4 (Original) The computer-implemented method of claim 1, wherein the first and second components are associated with separate servers.

5. (Original) The computer-implemented method of claim 1, wherein the first and second components are Enterprise Java Bean components.

6. (Original) The computer-implemented method of claim 5, wherein the first and second components are associated with a container.

7. (Original) The computer-implemented method of claim 6, further comprising placing the request from the first component is placed in a queue.

8. (Currently Amended) The computer-implemented method of claim 7, wherein ~~a~~the worker thread dequeues the received request after receiving a transaction commit signal from the container.

9. (Original) The computer-implemented method of claim 8, wherein the exception listener receives the exception and the scope of the exception.

10. (Previously Amended) A computer-implemented method for a first component to invoke a second component asynchronously in an object-oriented environment, the computer-implemented method comprising:

identifying a second component for handling a message;

transmitting a request associated with the message from a first component to invoke the second component;

registering an exception listener on an asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

11. (Original) The computer-implemented method of claim 10, wherein the asynchronous proxy has the same type as the second component.

12. (Original) The computer-implemented method of claim 10, wherein the request is associated with no application specific exceptions.

13. (Original) The computer-implemented method of claim 10, wherein the first and second components are associated with separate servers.

14. (Original) The computer-implemented method of claim 10, wherein the first and second components are Enterprise Java Bean components.

15. (Original) The computer-implemented method of claim 14, wherein the first and second components are associated with a container.

16. (Currently Amended) A~~The~~ computer program product comprising computer code for a first component to invoke a second component asynchronously, the computer readable medium comprising:

computer code for receiving a request from a first component to invoke a second component;

computer code for maintaining the scope of the received request;

computer code for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener for handling exceptions associated with the invocation of the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components;

a computer-readable medium for storing the computer codes.

17. (Original) The computer program product of claim 16, wherein the request has a return type of void.

18. (Original) The computer program product of claim 16, wherein the request is associated with no application specific exceptions.

19. (Original) The computer program product of claim 16, wherein the first and second components are associated with separate servers.

20. (Original) The computer program product of claim 16, wherein the first and second components are Enterprise Java Bean components.

21. (Original) The computer program product of claim 20, wherein the first and second components are associated with a container.

22. (Original) The computer program product of claim 21, further comprising placing the request from the first component is placed in a queue.

23. (Original) The computer program product of claim 22, wherein the worker thread dequeues the received request after receiving a transaction commit signal from the container.

24. (Original) The computer program product of claim 23, wherein the exception listener receives the exception and the scope of the exception.

25. (Previously Amended) A computer program product for a first component to invoke a second component asynchronously in an object-oriented environment, the computer program product comprising:

computer code for identifying a second component for handling a message;

computer code for transmitting a request associated with the message from a first component to invoke the second component;

computer code for registering an exception listener on an asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components;

a computer-readable medium for storing the computer codes.

26. (Original) The computer program product of claim 25, wherein the asycnhronous proxy has the same type as the second component.

27. (Original) The computer program product of claim 25, wherein the request is associated with no application specific exceptions.

28. (Original) The computer program product of claim 25, wherein the first and second components are associated with separate servers.

29. (Original) The computer program product of claim 25, wherein the first and second components are Enterprise Java Bean components.

30. (Original) The computer program product of claim 29, wherein the first and second components are associated with a container.

31. (Previously Amended) An enterprise environment associated with a computing system, the enterprise environment comprising:

an asynchronous proxy for receiving a request from a first component to invoke a second component;

an exception listener coupled to the asynchronous proxy, wherein the exception listener uses a scope corresponding to the request to handle exceptions associated with the invocation, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

32. (Original) The computer program product of claim 31, wherein the request has a return type of void.

33. (Original) The computer program product of claim 31, wherein the request is associated with no application specific exceptions.

34 (Original) The computer program product of claim 31, wherein the first and second components are associated with separate servers.

35. (Original) The computer program product of claim 31, wherein the first and second components are Enterprise Java Bean components.

36. (Original) The computer program product of claim 35, wherein the first and second components are associated with a container.

37. (Original) The computer program product of claim 31, wherein the worker thread invokes the second components after receiving a transaction commit signal from the container.

38. (Currently Amended) An enterprise environment associated with a computing system, the enterprise environment comprising:

~~memory containing a first component;~~

a processor coupled to~~with~~ memory, the processor configured to identify a second component for handling a message from a~~the~~ first component included in memory;

an interface coupled to~~with~~ the processor ~~and memory~~, the interface configured to transmit a request associated with the message from the first component to invoke the second component, wherein the interface also transmits information to register an exception listener on an asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

39. (Original) The computer-implemented method of claim 38, wherein the request has a return type of void.

40. (Original) The computer-implemented method of claim 38, wherein the request is associated with no application specific exceptions.

41. (Original) The computer-implemented method of claim 38, wherein the first and second components are associated with separate servers.

42. (Original) The computer-implemented method of claim 38, wherein the first and second components are Enterprise Java Bean components.

43. (Previously Amended) An apparatus for a first component to invoke a second component asynchronously in an object-oriented computing environment, the apparatus comprising:

means for receiving a request from a first component to invoke a second component;

means for maintaining the scope of the received request;

means for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener for handling exceptions associated with the invocation of the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

44. (Previously Amended) An apparatus for a first component to invoke a second component asynchronously in an object-oriented environment, the computer-implemented method comprising:

means for identifying a second component for handling a message;

means for transmitting a request associated with the message from a first component to invoke the second component;

means for registering an exception listener on an asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.